

2207/9865
PATENT

UNITED STATES PATENT APPLICATION
FOR

**METHOD AND APPARATUS FOR REDUCING MEMORY LATENCY
IN A CACHE COHERENT MULTI-NODE ARCHITECTURE**

INVENTORS:

Manoj Khare
Faye Briggs
Akhilesh Kumar
Lily Looi
Kai Cheng

PREPARED BY:

KENYON & KENYON
1500 K ST., N.W.
WASHINGTON, D.C. 20005

(202) 220-4200

003227-09364200

METHOD AND APPARATUS FOR REDUCING MEMORY LATENCY IN A CACHE COHERENT MULTI-NODE ARCHITECTURE

5 FIELD OF THE INVENTION

Embodiments of the present invention relate to a computer system having multi-node distributed shared memory architecture. The present invention relates to a method and apparatus for managing memory related requests and in particular for reducing memory latency in a multi-node architecture.

10 BACKGROUND

Computer systems may contain multiple processors that may work together to perform a task. For example, a computer system may contain four processors that may share system resources (e.g., input devices or memory devices) and may perform parallel processing. The processors may send messages to each other, may send messages to system resources, and may send and receive messages from the system resources. For example, such messages may include requests for information that is stored at a location in a memory device or a request to store information in a location of a memory device.

In a cache coherent shared memory multiprocessor, the set of data currently being used by a microprocessor may be copied from a system memory device such as a dynamic random access memory (DRAM) into a relatively smaller but faster cache memory device such as a static random access memory (SRAM). In such systems, a cache is said to be “coherent” if the information resident in the cache accurately reflects the information in DRAM memory. Lack of cache coherency can occur when a requester for a memory location does not receive the latest copy of the data. For example, if the cache is updated while main memory is not and a new requester for this location receives the data from main memory into the requester’s cache, the cache is said to be non-coherent.

Cache “snooping” is a technique used to detect an access to memory that might cause a cache coherency problem. For example, in the case where the cache is updated while the memory is not, the memory request from the new requesting agent is snooped in the cache containing updated data, which then supplies the data to the requester instead of memory.

5 In a multi-processor system, the messages sent between processors may include cache snooping messages generated in accordance with a coherence protocol. A coherence protocol (e.g., MESI protocol) is implemented to prevent cache coherency problems.

Typically, in order for the requester to receive updated data, the result of the cache coherence protocol is first determined before the memory access is started. In a large scale
10 multi-node distributed memory multi-processor, the resolution of the cache coherence protocol may take a long time, resulting in increased memory latency for accessing data from memory.

DESCRIPTION OF THE DRAWINGS

15 FIG. 1 is a partial block diagram of a system having a switching agent that manages memory requests according to an embodiment of the present invention.

FIG. 2 is a partial block diagram showing a memory address space for a MULTI-NODE system according to an embodiment of the present invention.

20 FIG. 3 is an exemplary snoop filter according to an embodiment of the present invention.

FIGS. 4 is a flow diagram of a method for reducing memory latency in accordance with embodiments of the present invention.

FIGS. 5 is a flow diagram of a method for reducing memory latency in accordance with embodiments of the present invention.

DETAILED DESCRIPTION

Embodiments of the present invention relate to methods and apparatus for reducing memory latency in a cache coherent multi-node architecture using speculative memory read commands.

5 In an embodiment of the invention, before the coherence protocol results are determined or completed, a requesting node or a coherence agent on behalf of the requesting node may issue a speculative memory read request to a home node of the requested memory location. The home node having the requested location may be defined as the node whose main memory stores the data for memory location (address) to be read. The home node that receives the speculative read request may access a memory address space to retrieve data specified by the speculative read request. While the home node of the memory location processes the speculative read request, the coherence agent determines the results of a cache coherence protocol. Based on these results, the coherence agent may send a cancel or confirm command to the home node. The cancel command causes the home node to drop the retrieved data, while the confirm command causes the home node to return the accessed data to the requesting agent.

10 The coherence protocol may require a snoop filter or directory look-up table located within the coherence agent, for example, to determine the location and status of the cached copies of the requested data. The snoop filter may determine whether the requested data is cached in an invalid, exclusive or shared state in one of the plurality of nodes. Based on the result of the coherence protocol, a “confirm” or “cancel” command may be issued to the home node. If a confirm command is issued, the home node of the memory location supplies the data to the coherence agent.

20 In alternative embodiments, the snoop filter may determine, as a result of plurality of requests and responses, that one of the other nodes in the multi-node system has a more

current copy of the requested data and/or can deliver it faster. Accordingly, a cancel command may be issued to the home node to drop the data accessed from memory and a read request may be sent to the other node to retrieve the requested data. In this case, the home node terminates the processing of the speculative read request and continues to process other requests.

Embodiments of the present invention reduce memory latency in a multi-node architecture. The present invention uses speculative read requests before results of a cache coherence protocol are determined. Speculative memory access is designed to be independent of the cache coherence protocol. Speculative memory requests are confirmed or cancelled after the results of the coherence protocol are determined. In embodiments of the present invention, since speculative memory reads can occur parallel to the processing of the cache coherence protocol, memory latency is reduced.

FIG. 1 is a partial block diagram of a system having a central coherence/switching agent that manages memory requests according to an embodiment of the present invention. FIG. 1 shows a system 100 which is a computer system that includes processors, memory devices, and input/output devices. Components in system 100 are arranged into architectural units that are referred to herein as "nodes." A system that has multiple nodes may be referred to as a multi-node system.

Each node may contain one or more processors, memories, and/or input/output devices. The components within a node may be connected to other components in that node through one or more busses and/or lines. In addition, each node in system 100 has an external connector that may be used by the components within that node to communicate with components in other nodes. In one embodiment, any communication from a component within a first node to another node must be routed though the external connection for the first node. In system 100, the external connection for each node is connected to a central coherence/switching agent 140 (hereinafter referred to as coherence agent 140). In an

alternative embodiment of the present invention, each of the nodes may communicate with other nodes through direct connections (not shown) that do not pass through coherence agent 140.

In embodiments of the present invention, the coherence agent 140 could be implemented in a central switch for all nodes (as shown in FIG. 1) or, alternatively, the coherence agent could be implemented in a distributed manner integrated in the node controllers of the node(s) (not shown).

The nodes in system 100 may send messages that are directed to a processor or resource in another node. For example, one node may send a request to read or write data to a memory location that is located in another node. Similarly, a node and/or coherence agent 140 may send a request to snoop the caches in the other nodes. In one embodiment, all requests in system 100 from one node to another node may be sent to coherence agent 140, and coherence agent 140 may send requests to other nodes that are based on the first request. For example, coherence agent 140 may receive a request from a first node to read or write data to a particular memory location, and coherence agent 140 may send snoop requests to the other nodes in system 100 as needed to carry out the received request from the first node. Alternatively, a node may send requests to another node without sending the same request to the coherence agent 140.

The details shown in FIG. 1 will now be discussed. As shown in FIG. 1, system 100 includes, for example, first node 110, second node 120, third node 130, and input/output node 150. Each of these nodes is coupled to coherence agent 140. The term “coupled” encompasses a direct connection, an indirect connection, an indirect communication, etc. First node 110 is coupled to coherence agent 140 through external connection 118, second node 120 is coupled to coherence agent 140 through external connection 128, and third node 130 is coupled to coherence agent 140 through external connection 138. External connections 118, 128, and 138 may be one or more lines capable of communicating

information to and from the node. In embodiments of the invention, the nodes may be coupled to each other through direct connections (not shown).

First node 110 includes processor 111, processor 112, and node controller 115, which are coupled to each other by bus 114. Processor 111 and processor 112 may be any micro-
5 processors that are capable of processing instructions, such as for example a processor in the INTEL PENTIUM family of processors. Bus 114 may be a shared bus. First node 110 also contains a memory 119 which is coupled to node controller 115. Memory 119 may be a Random Access Memory (RAM). Processor 111 may contain a cache 113, and processor 112 may contain a cache 117. Cache 113 and cache 117 may be Level 2 (L2) cache
10 memories that are comprised of SRAM. Of course, first node 110 may include processors additional to the ones shown (e.g., 111, 112).

Similarly, second node 120 contains processors 121, 122 and node controller 125 that are coupled to each other by bus 124. Second node 120 also contains a memory 129 that is coupled to node controller 125. Third node 130 contains processors 131, 132, and node
15 controller 135 that are coupled to each other by bus 134. Third node 130 also contains a memory 139 that is coupled to node controller 135. Processors 121, 122 may contain cache 123, 127, respectively. Processors 131, 132 may contain cache 133, 137 respectively. Processors 121, 122, 131, and 132 may be similar to processors 111 and 112. In an embodiment, two or more of processors 111, 112, 121, 122, 131, and 132 are capable of
20 processing a program in parallel. System 100 may include additional nodes, and each of the nodes (e.g., 110, 120, 130, etc.) may include processors additional to the ones shown (e.g., 111, 112, 121, 122, 131, 132). Node controllers 125 and 135 may be similar to node controller 115, and memory 129 and 139 may be similar to memory 119.

In an embodiment of the present invention, coherence agent 140 may be a routing
25 switch for managing speculative reads and/or writes and for routing messages within system 100. As shown in FIG. 1, coherence agent 140 contains a coherence controller 142 and a

snoop filter or directory look up table 143 that may be stored in memory 144. The snoop filter 143 may include a plurality of memory addresses as well as corresponding memory status indicators. The memory status indicators may indicate the status of data located in the plurality of corresponding memory addresses.

5 The coherence controller 142 may monitor reads and writes to memory locations (e.g., in cache and main memory) that may occur in system 100 and may update the snoop filter 143 as necessary. The coherence controller 142 may perform speculative reads to cache and/or main memory to reduce latency in a cache coherent multi-node architecture in accordance with embodiments of the present invention.

10 As shown in FIG. 1, input/output node 150 contains an input/output hub 151 that is coupled to one or more input/output devices 152. Input/output devices 152 may be, for example, any combination of one or more of a printer, keyboard, mouse, or any other input/output device. Input/output hub 151 may be an integrated circuit that contains bus interface logic for interfacing with a bus that complies to, for example, the Peripheral Component Interconnect (PCI) standard. PCI Special Interest Group (SIG) PCI Local Bus Specification, Revision 2.2, published December 18, 1998. Input/output hub 150 may be similar to, for example, the INTEL 82801AA I/O Controller Hub.

15 In an embodiment, node controller 115, coherence agent 140, and input/output hub 151 may be a chipset that provides the core functionality of a motherboard, such as a modified version of a chipset in the INTEL 815 family of chipsets.

20 In a further embodiment, inter-node communication in system 100 may be asynchronous (i.e., there is no fixed timing between events). In a still further embodiment, inter-node communication may be sent in the form of packets which may contain a header or a header and data sections. An example of a message size may be 144 bits. In an embodiment, the messages sent may include requests and responses. In a further embodiment, the types of requests that the nodes may send and receive may include a

memory read request, memory write request, cache snoop request, cache flush request, memory update request, cache line replacement request, input/output port read request, and input/output port write request. Requests may contain fields such as a packet type, destination ID, request type, source ID, transaction address, request length, stream ID, and ordering semantics.

In an embodiment of the present invention, the processors in nodes 110, 120 and 130 may be shared memory multi-processors and each of the memories 119, 129 and 139 may be part of the same shared physical address space. In a further embodiment, the processors in nodes 110, 120, and 130 communicate with each other through shared memory reads and writes (i.e., by writing to and reading from memory 119, 129 and 139). In a further embodiment, the processors in nodes 110, 120 and 130 each have one or more caches (e.g., Level 1 and Level 2 caches) and these caches are kept coherent using coherence agent 140. For example, when processor 111 accesses a location in memory 119, it may send a snoop request for that memory location to coherence agent 140, which may determine if any of the processors in second node 120 and third node 130 have cached that memory location. A snoop request may be generated when a processor needs other processors in the system to look in their own caches to see if a particular line is present in their cache.

FIG. 2 is a partial block diagram showing a memory address space 201 and caches for a multi-node system according to an embodiment of the present invention. Memory address space 201 includes first memory 119, second memory 129, and third memory 139, which are shown respectively in first node 110, second node 120, and third node 130 in FIG. 1. For the purpose of illustration, first memory 119 is shown as containing the addresses 0 to 49, second memory 129 is shown as containing the addresses 50 to 99, and third memory 139 is shown as containing the addresses 100 to 149. Thus, in this embodiment, address space 201 contains the locations in addresses 0 to 149. Of course, in other embodiments address space 201 may contain other addresses and address space 201 may be spread across

more or less memories than as shown in FIG. 2. Address space 201 may be referred to as a “shared” address space because any node in the multi-node system may address any location in address space 201. Thus, second node 120 may send a request to read from location 0 in the address space, location 0 being contained in memory 119 in first node 110.

FIG. 2 also shows caches 113, 123, and 133, which are contained in first node 110, second node 120, and third node 130 of FIG. 1. The multi-node system may contain more or less caches than is shown in FIG. 2 (such as caches 117, 127 and 137 of FIG. 1). In an embodiment, each addressable location of memory in address space 201 may be the size of a cache line in caches 113, 123, and 133. In FIG. 2, certain locations in the memory address space 201 are designated as A, B, C, D, E and F for the purposes of reference. Each of these locations stores some information (data or instructions). The information in any particular location in the memory address space 201 may be cached in one or more of the caches in the system. Thus, a copy of the information in the location designated A in first memory 119 is cached in caches 113, 123, and 133; a copy of the information in the location designated B in first memory 119 is cached in cache 123; a copy of the information in the location designated C in first memory 119 is cached in caches 113 and 133; a copy of the information in the location designated D in second memory 129 is cached in cache 123 and cache 133; a copy of the information in the location designated E in second memory 129 is cached in cache 113; and a copy of the information in the location designated F in third memory 139 is cached in cache 133.

Coherence agent 140 may process requests as follows. If coherence agent 140 receives from first node 110 a request to access the memory location designated as D (in FIG. 2), the coherence controller 142 may determine from snoop filter 143 that memory location D is cached in both cache 123 (in second node 120) and cache 133 (in third node 130). Cache coherence controller 142 may then cause snoop requests that are associated with

location D to be sent to second node 120 and third node 130 to access the memory location designated as D.

FIG. 3 is an illustration of an exemplary snoop filter or directory look-up table 143 in accordance with embodiments of the invention. Snoop filter 143 may be stored in memory 144 and may include information that may be regularly accessed and updated by the coherence controller 142. Snoop filter 143 may include rows and columns for storing, for example, memory address locations 301, an identifier for the node and corresponding processor where the data is located 303 and caching or snoop status 304. Unless otherwise specified, memory as referred to herein may refer to cache or main memory. Memory location 301 may contain memory addresses of a plurality of memory locations 0 through N that may be “snooped” by the coherence controller 142 (e.g., 0 through 149 discussed above). The column for node/processor identifier 303 may contain the identity of the node and corresponding processors that have data stored in memory (where X is, for example, any node in system 100 and Y is any processor in that node having the corresponding data).

The column for memory status 304 indicates the status of the corresponding data stored in the memory as specified by memory address 301. The memory status may indicate whether, for example, the stored data is in an invalid state (I), exclusive state (E) or shared state (S). An invalid state indicates that none of the nodes in the system 100 have the memory location identified by the memory address cached (i.e., a copy of the data stored in a cache). An exclusive state indicates the memory location identified by the memory address may be cached exclusively in only one node in the system. It is possible that the node with the exclusive copy of the data may have modified the data without informing the coherence controller 142. If the data has been modified, the data is considered to be “dirty” (i.e., in a modified state). The shared state indicates that one or more nodes in the system may have cached the data in a “clean” state (i.e., unmodified state).

The snoop filter 143 may be stored in an internal memory within the coherence agent 140 or in an external memory 144. The coherence controller 142 may read and write to snoop filter 143 when, for example, snooping one of the nodes in system 100 or performing other read and write requests.

FIGS. 4 and 5 are flow diagrams relating to a method for reducing memory latency in a cache coherent multi-node architecture using speculative memory read commands, according to embodiments of the present invention.

Referring to FIG. 4, a requesting node, for example, first node 110 may issue a read request to request data from another node (401). The read request includes the memory address of the memory location to be read. A requesting node may be any of the plurality of nodes of system 100. In an embodiment of the invention, the requesting node sends the read request to a coherence agent, for example, agent 140. Upon receiving the read request, coherence agent 140 may immediately issue a speculative read request to a home-node (403). The speculative read request is issued before coherence protocol results are determined or retrieved. The home node is identified as the node that contains the memory address of the memory location to be read (i.e., the “home” of the data to be read).

In alternative embodiments of the invention, the requesting node may directly issue the speculative read request including a memory address to the home node. If the requesting node issues the speculative read request directly, the requesting node also sends a read request to the coherence agent. The read request may be sent simultaneous with or soon after the speculative read request. The term requester as used herein may refer to either a requesting node or a coherence agent.

After the home node receives the speculative read request from either the requesting node or the coherence agent, the home node initiates a read to main memory (405). The speculative read request is received before coherence protocol results are received. Thus, for example, if node 130 shown in FIG. 1 is the home node, home node 130 initiates a read to

main memory 139. Node controller 135, of the home node 130, accesses the memory location specified by the speculative read request. The node controller determines whether the speculatively read data can be buffered by the home node (407). The node controller determines whether buffering the speculatively read data will cause a buffer full condition that will prevent it from supporting, for example, non-speculative memory requests. If a buffer full condition occurs, data that has been speculatively read may be dropped. Speculative data that has been buffered for a longer period of time may be dropped first. If the speculatively read data can be buffered, the home node buffers the data (409). If the speculatively read data can not be buffered, the speculatively read data is dropped (417).

After the speculatively read data is buffered, the node controller of the home node awaits a confirm or cancel command relating to the speculatively read data. The confirm or cancel command is sent by a coherence controller of a coherence agent after results of a cache coherence protocol are determined (to be described below in more detail). If a cancel command is received, the speculatively read data is dropped (411 and 417). If a confirm command is received and the speculatively read data still exists in the buffer, the data is forwarded to the coherence agent (411, 413 and 415).

If a confirm command is received and the data no longer exists in the buffer, the node controller may read the requested data specified from main memory (411, 413 and 419). In embodiments of the invention, if the data is read from main memory, the data is buffered before it is forwarded to the coherence agent. After the data is read, the data is forwarded to the coherence agent (415).

After the coherence agent receives the data, the coherence agent forwards the data to the requesting node, for example, node 110 (421). In alternative embodiments, the home node may forward the data directly to the requesting node.

In embodiments of the invention, the cancel and confirm commands as well as the read and speculative read requests are issued with the full address of the requester and the

transaction ID of the transaction issued by the requester to uniquely identify the command and/or request.

Referring to FIG. 5, after the requesting node issues a read request to the coherence agent, the coherence agent, for example, coherence agent 140 initiates a cache coherence resolution procedure in accordance with embodiments of the present invention (501). As described above, if the requesting node has not issued the speculative read request directly to the home node, the coherence agent issues the speculative read request to the home node. The speculative read request may be issued at or about the same time as the initiation of the coherence resolution protocol, but before the results of the cache coherent protocol are determined. The read request and the speculative read request includes the memory address of the memory location to be read.

According to embodiments of the present invention, the cache coherence resolution procedure may be initiated by coherence controller 142. The coherence controller 142 looks-up the memory address in the snoop filter table 143 and determines the caching status of the memory location (503). If the memory location is determined to be in an invalid or shared state, the coherence controller issues a confirm command to the home node (505 and 507).

The coherence controller updates the snoop filter with the updated caching status. The confirm command is issued with the full address of the memory location of the requested data. After receiving the confirm command, the node controller of the home node determines whether the data still exists in a buffer (509). If so, the data is forwarded to the coherence agent (511). If the data is no longer in the buffer, the data is read from the main memory and forwarded to the coherence agent (513 and 511). The coherence agent forwards the requested data to the requesting node, for example, node 110 (515).

If in step 505, the memory location looked up by the coherence controller is determined to be in an exclusive state, the coherence controller snoops the node having an exclusive copy of the desired data cached (517). A snoop response is received that includes

the status of the data as well as a copy of the snooped data if the data is modified (i.e., dirty). After the coherence controller receives the snoop response, the controller determines whether the corresponding data is “clean” or “dirty” (519). As indicated above, the “clean” state indicates that the data has not been modified while the “dirty” state indicates that the data has been modified. If the data is clean, then a confirm command is issued to the home node (507). The coherence controller updates the snoop filter with the updated caching status. The home node then retrieves the data and forwards the data to the coherence agent (509-513). The coherence agent forwards the requested data to the requesting node (515).

If the data is “dirty,” the coherence agent may issue a cancel command to the home node (521). The coherence controller updates the snoop filter with the updated caching status. The cancel command is issued with the full address of the memory location of the requested data. In response to the cancel command, the home node drops the speculatively read data (523). In embodiments of the present invention a cancel command to the home node may not be required since the speculatively read data will be automatically dropped when the buffer can no longer keep the data (e.g., on a buffer full condition). In further alternative embodiments, the coherence agent may issue a write request to the home node. The write request may be issued if the requested memory location is found to be in, for example, a modified state in another node and the modified data may be reflected back to the home node. Thus, in this case the write request to update the home node could act as the cancel command that causes the speculatively read data to be dropped.

After or about the same time the cancel command to the home node is issued, the coherence controller forwards the snooped data to the requesting node (525). In an alternative embodiment, the node having the “dirty” or modified data may forward the data directly to the requesting node.

Embodiments of the invention may reduce memory latency in a multi-node architecture. The present invention uses speculative read requests before results of a cache

coherence protocol are known. Speculative memory access is independent of the cache coherence protocol. Speculative memory requests are confirmed or cancelled after the results of the coherence protocol are determined. Embodiments of the invention, may reduce memory latency since speculative memory reads can occur while of the cache coherence protocol is being processed.

Several embodiments of the present invention are specifically illustrated and/or described herein. However, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention. For example, while the nodes in FIG. 1 are shown containing one or two processors, a node may contain any number of processors. In one embodiment, a node contains only one processor and in another embodiment a node contains sixteen processors. As another example, while the nodes in FIG. 1 are connected through coherence agent 140, in another embodiment two nodes may be directly connected to each other. For example, in a system with that has two nodes, the node controller of a first node may be connected to the node controller of a second node.

In another embodiment, the node controller for a node may be part of a processor in that node. For example, a first node in a system may only contain one processor and the node controller for that node may be part of that processor. In another embodiment, the coherence agent and the associated coherence controller and snoop filter or directory, may be distributed. The distributed switch might in turn be integrated as part of the node controller. In addition, a node (e.g., node 110) may contain one or more processors and an input/output hub. In still another embodiment, the snoop filter, read/write requests and/or other commands are implemented as instructions adapted to be executed by a processor, which may include firmware, microcode, object code, source code, etc.

